Razonamiento Temporal Métrico

María Laura Cobo Marcelo A. Falappa
Departamento de Ciencias e Ingeniería de la Computación *
Universidad Nacional del Sur
Bahía Blanca - Argentina
fax: +54 291 4595136
[mlcobo, mfalappa] @cs.uns.edu.ar

Resumen

Durante las últimas décadas la investigación de temas relacionados con la representación y manipulación de conceptos como tiempo y cambio han adquirido relevancia. En particular se han tomado diversas lógicas que capturan estos conceptos y han sido reutilizadas en Ciencias de la Computación.

Las lógicas en general fueron desarrolladas desde la filosofía para capturar el comportamiento y uso del tiempo en las lenguas vivas. Muchas de estas lógicas resultan de interés para los propósitos requeridos en algunos sistemas de cómputo.

En los últimos años, han aparecido muchas propuestas para contar con lenguajes temporales, en particular se ha extendido esa lista con dos lenguajes temporales métricos: $MTPL$ [CA00] y $EMTPL$ [CA99]. Ambos lenguajes utilizan como fundamento la lógica métrica de Prior [Pri67a], la cual ha sido profundamente estudiada desde la practica.

Al momento de su desarrollo estos lenguajes fueron pensados para la especificación de propiedades. La principal idea detrás de este trabajo es indagar sobre la aplicabilidad de ese lógicas en razonamiento temporal o extender los lenguajes de manera que sean aptos para un entorno de razonamiento temporal

Palabras Clave: Lenguajes de programación y especificación, Lógica Temporal Métrica, Negación, Razonamiento Temporal.

1. Introducción

El Razonamiento Temporal [Vil94] consiste de la formalización de la noción de tiempo y proveer un medio para representar y razonar con los aspectos temporales del conocimiento. Los sistemas que se han desarrollado para razonar con tiempo pueden dividirse básicamente en dos grupos:

Basados en lógicas de primer orden: En general este tipo de sistemas agrega el tiempo en la forma de un parámetro adicional, que normalmente recibe el nombre de

*Parcialmente financiado por la Secretaría de Ciencia y Tecnología (Universidad Nacional del Sur).

argumento temporal. La principal ventaja de esta aproximación radica en la capacidad de poder utilizar todos los mecanismos de razonamiento diseñados para la lógica de primer orden.

Basados en operadores modales: En este caso se enriquece el cálculo propocisional o de primer orden con operadores temporales. La mayoría de los desarrollos en este sentido utilizan operadores no métricos, más específicamente, los operadores clásicos:

- $\Diamond\!\!\!\!+\ \phi$ alguna vez en el futuro ϕ o ϕ será verdadera en algún momento futuro.
- $\Diamond\ \phi$ alguna vez en el pasado ϕ o ϕ fue verdadera en algún momento pasado.
- $\boxplus\ \phi$ siempre en el futuro ϕ o ϕ será verdadera en todos los momentos futuros.
- $\boxminus\ \phi$ siempre en el pasado ϕ o ϕ fue verdadera en todos los momentos pasados.

En general se aduce que la resolución en sistemas basados en esta lógica es mucho menos eficiente que en el caso anterior. De todos modos algunos investigadores, están trabajando en métodos de resolución para este tipo de lógicas y más aún están tratando de mejorar la eficiencia de los mismos.

Un aspecto que puede llamar la atención es que no hay intentos para lograr razonamiento con operadores temporales métricos, aunque si los hay para lógicas basadas en intervalos y para lógicas métricas basadas en lógicas con restricciones. Nuestro objetivo es poder adaptar los lenguajes $MTPL$ o $EMTPL$ para que resulten aptos en un entorno donde se requiera algún grado de razonamiento temporal.

2. Breve revisión a $MTPL$ y $EMTPL$

Los lenguajes de programación $MTPL$ [CA00] y $EMTPL$ [CA99] están basados en la lógica métrica de Prior [Pri67a]. $MTPL$ captura una forma de tiempo en la cual hay un momento destacado y todas las referencias temporales de cualquier fórmula hacen referencia a ese momento destacado. En cambio, en el caso de $EMTPL$ se logra establecer una combinación entre esta manera de representar el tiempo y la opuesta, en la cual las referencias son independientes de cualquier momento de tiempo.

2.1. Sintaxis y semántica

La sintaxis de los operadores que utilizan MTPL y EMTPL es la siguiente:

$$\phi = p|\neg\phi|\phi_1 \wedge \phi_2|AT_n\phi|\Diamond\!\!\!\!+_n\ \phi|\exists n\ \Diamond\!\!\!\!+_n\ \phi|\forall n\ \Diamond\!\!\!\!+_n\ \phi|\Diamond_n\ \phi|\exists n\ \Diamond_n\ \phi|\forall n\ \Diamond_n\ \phi$$

La disyunción e implicación puede ser definidas de la forma usual. A partir de este conjunto de operadores pueden definirse otros. Teniendo en cuenta el significado intuitivo presentado anteriormente, se puede ver cual es la semántica formal de los operadores. Se asume un conjunto de instantes $\ldots, i_n, i_{n+1}, i_{n+2}, \ldots$ que representan una estructura temporal lineal e ilimitada. La información en el sistema puede verse como una lista de conjuntos, σ, que contiene todas las verdades que tiene la teoría. Cada miembro de la lista es denotado como $\sigma(j)$. Luego $\sigma(j)$ representa el conjunto de verdades en un determinado instante j. A continuación, puede verse una definición inductiva para "una proposición

temporal verdadera en un instante j". Se comienza especificando que $p \in \sigma(j)$ significa "p es verdadera en el instante j".

Es importante destacar que la cuantificación es solo aplicada a miembros de la estructura temporal, es decir, el conjunto de números que se escogieron como representación del tiempo. Se exige que este conjunto de números conforme un espacio métrico.

$(\sigma, j) \models p$ si solo si $p \in \sigma(j)$
$(\sigma, j) \models \neg p$ si solo si $(\sigma, j) \not\models p$
$(\sigma, j) \models p \wedge q$ si solo si $(\sigma, j) \models p$ y $(\sigma, j) \models q$
$(\sigma, j) \models \oplus_c\ p$ si solo si $(\sigma, j + c) \models p$
$(\sigma, j) \models \ominus_c\ p$ si solo si $(\sigma, j - c) \models p$
$(\sigma, j) \models \exists n\ \oplus_n\ p$ si solo si existe un instante $n,\ n > 0$ tal que $(\sigma, j + n) \models p$
$(\sigma, j) \models \exists n\ \ominus_n\ p$ si solo si existe un instante $n,\ n > 0$ tal que $(\sigma, j - n) \models p$
$(\sigma, j) \models \forall n\ \boxplus_n\ p$ si solo si para todo instante $n,\ n > 0$ se tiene $(\sigma, j + n) \models p$
$(\sigma, j) \models \forall n\ \boxminus_n\ p$ si solo si para todo instante $n,\ n > 0$ se tiene $(\sigma, j - n) \models p$
$(\sigma, j) \models AT_c\ p$ si solo si $(\sigma, c) \models p$

Se pueden definir otros operadores interesantes a partir de éstos [CA00, CA99]

2.2. Definición del lenguaje

Los lenguajes están basados en la lógica de Prior [Pri67a, Pri67b] más una extensión con la introducción del operador AT en el caso del lenguaje $EMTPL$. La extensión consiste principalmente en la introducción del operador AT en la axiomatización de Prior [CA99].

La lógica utilizada entonces cuenta con todas las reglas y teoremas del cálculo proposicional, además de una teoría de cuantificación sobre las referencias métricas que se realizan en los operadores temporales. Junto a los siguientes esquemas de axiomas:

Ax1 :	$\oplus_n\ (p \to q) \to (\oplus_n\ p \to \oplus_n\ q)$	Ax1′ :	$\ominus_n\ (p \to q) \to (\ominus_n\ p \to \ominus_n\ q)$
Ax2 :	$\oplus_n \ominus_n\ p \to p$	Ax2′ :	$\ominus_n \oplus_n\ p \to p$
Ax3 :	$\oplus_m\ \exists n\ \oplus_n p \to \exists n\ \oplus_m \oplus_n\ p$	Ax3′ :	$\ominus_m \exists_n\ \ominus_n\ p \to \exists n\ \ominus_m \ominus_n\ p$
Ax4 :	$\oplus_m\ \exists n\ \ominus_n p \to \exists n\ \oplus_m \ominus_n\ p$	Ax4′ :	$\ominus_m \exists n\ \oplus_n\ p \to \exists n\ \ominus_m \oplus_n\ p$
Ax5 :	$\oplus_{(m+n)}\ p \to \oplus_m \oplus_n\ p$	Ax5′ :	$\ominus_{(m+n)}\ p \to \ominus_m \ominus_n\ p$
Ax6 :	$\oplus_m \oplus_n\ p \to \oplus_{(m+n)}\ p$	Ax6′ :	$\ominus_m \ominus_n\ p \to \ominus_{(m+n)}\ p$
Ax7 :	$\oplus_n\ \neg p \to \neg \oplus_n\ p$	Ax7′ :	$\ominus_n\ \neg p \to \neg \ominus_n\ p$
Ax8 :	$\oplus_n\ p \vee \oplus_n\ \neg p$	Ax8′ :	$\ominus_n\ p \vee \ominus_n\ \neg p$

y las reglas de inferencia

si $\vdash \alpha$ entonces $\vdash \oplus_n\ \alpha$ si $\vdash \alpha$ entonces $\vdash \ominus_n\ \alpha$

La definición de ambos lenguajes está dada por un subconjunto de las fórmulas bien formadas de la lógica de origen.

Definición **1** (MTPL)

Se tiene en consideración un lenguaje, MTPL, con átomos proposicionales, los conectivos lógicos: $\wedge$,$\vee$ $\to$, $\neg$ y los operadores temporales: $\oplus_n$, $\ominus_n$, $\oplus$, $\ominus$, $\boxplus$, $\boxminus$. Se definirán las nociones de *programa, cláusula, cláusula always, cláusula ordinaria, cabeza, cuerpo* y *meta*

- Un *programa* es un conjunto de "cláusulas".
- Una *cláusula* es o bien una "cláusula always" o bien es una "cláusula ordinaria".
- Una *cláusula always* es $\boxplus A$ o $\boxminus A$, donde A es una "cláusula ordinaria".
- Una *cláusula ordinaria* es una "cabeza de cláusula" o $B \rightarrow H$, donde B es un "cuerpo de cláusula" y H es una "cabeza de cláusula".
- Una *cabeza de cláusula* es una fórmula atómica, $\diamondsuit_n A$ o $\diamondsuit_n A$, donde A es una conjunción de "cláusulas ordinarias".
- Un *cuerpo de cláusula* es una formula atómica, una conjunción de cuerpos de cláusula, $\diamondsuit_n B$ o $\diamondsuit_n B$ o $\neg B$ donde B es un cuerpo de cláusula.
- Una *meta* es cualquier "cuerpo de cláusula" o disyunción de "cuerpos de cláusula".

El lector puede encontrar más detalles de este lenguaje en [CA00].

Definición **2** (EMTPL)
Se considera un lenguaje con átomos proposicionales, los conectivos lógicos: $\wedge$, $\vee$ $\rightarrow$, $\neg$ y los operadores temporales: $\diamondsuit_n$, $\diamondsuit_n$, $\diamondsuit$, $\diamondsuit$, $\boxplus$, $\boxminus$. Este lenguaje se puede definir como una extensión del lenguaje $MTPL$. Básicamente se modifica la definición de *cláusula* y *meta* de la siguiente manera:

- Una *cláusula* es $AT_i C$ o C donde C es una "cláusula ordinaria".
- Una *meta* es $AT_n B$ o B donde B es cualquier cuerpo de cláusula o disyunción de cuerpos de cláusula.

Intuitivamente la idea detrás del lenguajes $EMTPL$ era contar con un lenguaje temporal métrico que fuera apropiado para su uso en un contexto de base de datos deductivas. En ese sentido $MTPL$ presenta un serio problema de actualización por contar con un momento destacado de tiempo, normalmente llamado `presente`. Este problema hace que cada vez que se modifica el punto de referencia (`presente`) todas las referencias de la base deban modificarse, hecho que conlleva un tiempo de actualización prohibitivo. Con la incorporación del operador AT_i se logra evitar ese costo, a su vez su incorporación resulta en lenguajes temporales ya que logra, junto a los otros operadores temporales, capturar lo mejor de las dos concepciones temporales más conocidas (Series A y B según la descripción de McTaggart [Pri67a, Pri67b]) . El lector puede encontrar más detalles del lenguaje en [CA99].

2.3. El algoritmo

El algoritmo que se presentará a continuación puede ser utilizado para una implementación de un intérprete para el lenguaje $MTPL$ [CA00]. Este algoritmo está basado en la noción de *árbol de computación etiquetado* (noción propuesta por Gabbay [Gab87]) pero tiene importantes modificaciones para adaptarlo al lenguaje métrico propuesto. A fin se simplificar la presentación del algoritmo se utilizarán las siguientes equivalencias:

$$\diamondsuit p \equiv \exists n \diamondsuit_n p \qquad \boxplus p \equiv \forall n \diamondsuit_n p$$

DEFINICIÓN **3** (adaptada de la dada en [Gab87])
Sea P una base de datos finita y sea G una meta. Se define la noción de *árbol de computación etiquetado* para éxito o falla finita de la meta G a partir de P. Si G tiene éxito se escribe P?G=1 y si G falla finitamente se escribe P?G=0.

Sea $(T, \leq, 0, V)$ un *árbol de computación etiquetado*, tal que $0 \leq t$ para todos los $t \in T$. V es una función de etiquetado que le asigna a cada $t \in T$ una tripleta $(\mathrm{P}(t), G(t), X(t))$, donde $\mathrm{P}(t)$ es la base de conocimiento en el momento t, $G(t)$ es la meta en ese momento t y $X(t)$ es el propósito para ese meta. Si $X(t) = 0$ el propósito de la consulta es que la meta sea respondida de una manera negativa; en cambio si $X(t) = 1$, el propósito de la computación es el éxito de la misma.

Una observación importante radica en el hecho de que P es la implementación de σ es la lógica, es decir que $\mathrm{P}(j)$ toma el rol de $\sigma(j)$ en la semántica. Como P es siempre un conjunto finito, $\mathrm{P}(j)$ también lo es para cada j.

Como se hará referencia a ella, vale la pena recordar que la *regla del espejo* consiste en el cambio de todos los operadores del pasado por sus correspondientes operadores del futuro y viceversa. A través de su uso se cambia $\diamondminus_n$ por $\diamondplus_n$, $\diamondminus$ por $\diamondplus$ y $\boxminus$ por $\boxplus$ y lo mismo al revés.

El algoritmo que se presenta a continuación tiene como entradas la meta G, un propósito representado por la variable X y un programa P en lógica temporal métrica. La salida del algoritmo es la respuesta de satisfacción o falla de la meta de acuerdo al objetivo indicado para ella con respecto a P.
ALGORITMO PARA $MTPL$ (adaptado de [Gab87]) $(T, \leq, 0, V)$ es un árbol de computación etiquetado para P?$G = x$ si solo si se satisfacen las siguientes condiciones:

I. $V(0) = (P, G, x)$

II. Si $t \in T$ es un punto final

   a) $X(t) = 1$,$G(t)$ es un átomo q y $q \in \mathrm{P}(t)$ o $\diamondplus_0\, q \in \mathrm{P}(t)$ o $\diamondminus_0\, q \in \mathrm{P}(t)$

   b) $X(t) = 0$

      1) $G(t)$ es un átomo q y q no es la cabeza de ninguna cláusula ordinaria.
      2) $G(t)$ tiene la forma $\diamondplus_n\, A$ (o $\diamondminus_n\, A$) y no hay cláusulas con cabezas de la forma $\diamondplus_n\, D$ (respectivamente $\diamondminus_n\, D$).

III. Si $t \in T$ no es un punto final

   a) (*)[1] $G(t) = \neg A$, entonces t tiene exactamente un sucesor inmediato s, donde $\mathrm{P}(s) = \mathrm{P}(t)$, $X(s) = 1 - X(t)$ y $G(s) = A$

   b) $X(t) = 1$

      1) $G(t)$ es un átomo q entonces t tiene exactamente un sucesor inmediato s en el árbol con $\mathrm{P}(s) = \mathrm{P}(t)$, $X(s) = 1$ y $G(s) \rightarrow q \in \mathrm{P}(t)$
      2) $G(t) = A_1 \wedge A_2$, entonces t tiene exactamente dos sucesores inmediatos s_1 y s_2 en al árbol con $\mathrm{P}(s_i) = \mathrm{P}(t)$, $X(s_i) = 1$ y $G(s_i) = A_i$
      3) $G(t) = \diamondplus_n\, A$ entonces ir a `Caso Métrico:`$\diamondminus_n$
      4) $G(t) = \diamondplus A$, entonces entonces ir a `Caso No Métrico:`$\diamondminus$

[1] Esta regla del algoritmo será modificada en las extensiones para el lenguaje sugeridas en este trabajo.

5) $G(t) = A_1 \vee A_2$, entonces t tiene exactamente un sucesor inmediato: s con $\mathrm{P}(s) = \mathrm{P}(t)$, $X(s) = 1$ y $G(s)$ es o bien A_1 o bien A_2.

c) $X(t) = 0$

1) $G(t)$ es un átomo q, entonces t tiene $s_1, \ldots s_n$ como sucesores inmediatos en el árbol con $\mathrm{P}(s_i) = \mathrm{P}(t)$, $X(s_i) = 0$, y para algún $0 \leq k \leq n$ las cláusulas $G(s_i) \rightarrow q$ para $i \leq k$ son exactamente todas las cláusulas de la forma descripta en $\mathrm{P}(t)$

2) $G(t) = A_1 \wedge A_2$, entonces t tiene exactamente un sucesor inmediato s, $\mathrm{P}(s) = \mathrm{P}(t)$, $X(s) = 0$ y para algún $i \in 1, 2$ $G(s) = A_i$

3) Si t no es punto final, $X(t) = 0$ y $G(t) = \diamondsuit_n A$, entonces t tiene exactamente un sucesor inmediato: s y la siguiente condición se satisface: $\mathrm{P}(s)$ es el mismo conjunto que el expresado en III$b3$, $X(s) = 0$ y $G(s) = A$.

4) Si t no es un punto final, $X(t) = 0$ y $G(t) = \diamondsuit A$, entonces t tiene k sucesores inmediatos, $s_1, \ldots, s_k$ y para m, n donde siempre $1 \leq m \leq n \leq k$ y el conjunto de fbb (fórmulas bien formadas) $D_1, \ldots, D_k$, $D_i (i \leq m)$ donde estas son todas las cláusulas de la forma $\boxplus(B_i \rightarrow H_i)$ in $\mathrm{P}(t)$, $D(m+1), \ldots, D(n)$ son exactamente todas las cláusulas de la forma $\boxplus(B_i \rightarrow FH_i)$ in $\mathrm{P}(t)$, $D(n+1), \ldots, D(k)$ son exactamente todas las cláusulas de la forma $B_i \rightarrow \diamondsuit H_i$ in $\mathrm{P}(t)$, y para cada $1 \leq i \leq k$ una de las siguientes condiciones de satisface:

a' $\mathrm{P}(s_i) = \mathrm{P}(t)$, $X(s_i) = 0$ y $G(s_i) = B_i$

b' $\mathrm{P}(s_i) = \{\boxplus C | \boxplus C \in \mathrm{P}(t)\} \cup \{H_i\} \cup \{\diamondsuit C | C$ es una CO in $\mathrm{P}(t)\} \cup \{C | \diamondsuit_n C$ es una CO in $\mathrm{P}(t)\}$, $X(s_i) = 0$, $G(s_i) = A \vee \diamondsuit A$, para $1 \leq i \leq k$, excepto cuando $1 \leq i \leq m$, en cuyo caso $G(s_i) = \diamondsuit A$

5) $G(t) = A_1 \vee A_2$, entonces t tiene exactamente dos sucesores inmediatos: s_1 y s_2 con $\mathrm{P}(s_i) = \mathrm{P}(t)$, $X(s_i) = 0$ y $G(s_i) = A_i$.

Solo se presentaron las reglas correspondientes a los operadores del futuro, ya que las reglas correspondientes a los operadores del pasado son análogas. El algoritmo correspondiente a $EMTPL$ puede encontrarse en [CA99] no fue incluido por razones de espacio. oportunamente se probaron propiedades importentes sobre los algoritmos tales como correctituf y terminación [Cob01]. Las subrutinas utilizadas son las siguientes:

Caso No Métrico:$\diamondsuit$: Si t no es un punto final, $X(t) = 1$ y $G(t) = \diamondsuit A$, entonces t tiene un único sucesor, s_1 y se satisface la siguiente condición: $\mathrm{P}(s) = \mathrm{P}(t)$, $X(s) = 1$ y $G(s) = \diamondsuit_n A$

o tiene dos sucesores inmediatos, s_1 y s_2, y se satisface alguna de las siguientes condiciones:

1. $\mathrm{P}(s_1) = \mathrm{P}(t)$, $X(s_1) = 1$ y para algún H: $G(s_1) \rightarrow \diamondsuit H \in \mathrm{P}(t)$, $X(s_2) = 1$, $G(s_2) = A \vee \diamondsuit A$ y

$$\mathrm{P}(s_2) = \{\boxplus C | \boxplus C \in \mathrm{P}(t)\} \cup \{H\} \cup \{\diamondsuit C | C \text{ es una } \textit{cláusula ordinaria} \text{ in } \mathrm{P}(t)\}$$

2. análogo al item 1 reemplazando $G(s_1) \rightarrow \diamondsuit H \in \mathrm{P}(t)$ por $\boxplus(G(s_1) \rightarrow \diamondsuit H) \in \mathrm{P}(t)$
3. análogo al item 2 reemplazando $\boxplus(G(s_1) \rightarrow \diamondsuit H) \in \mathrm{P}(t)$ por $\boxplus(G(s_1) \rightarrow H) \in \mathrm{P}(t)$

Caso Métrico:$\diamondsuit_n$: Si t no es un punto final, $X(t) = 1$ y $G(t) = \diamondsuit_n A$, entonces:

- t tiene exactamente un sucesor inmediato, s, y la siguiente condición se satisface: $X(s) = 1$, $G(s) = A$ y

$$\begin{aligned}\mathrm{P}(s) = \quad & \{C|\Diamond_n\, C \in \mathrm{P}(t)\} \cup \{C|\boxplus C \in \mathrm{P}(t)\} \cup \{\boxplus C|\boxplus C \in \mathrm{P}(t)\} \cup \\ & \{\Diamond_n \boxminus C|\boxminus C \in \mathrm{P}(t)\} \cup \\ & \{\Diamond_n\, C|C \in \mathrm{P}(t) \text{ y } C \text{ es una cláusula ordinaria}\} \cup \\ & \{\Diamond_{(n-m)}\, C|\Diamond_m\, C \in \mathrm{P}(t) \text{ y } n > m\} \cup \{\Diamond_{(m+n)}\, C|\Diamond_m\, C \in \mathrm{P}(t)\} \cup \\ & \{\Diamond_{(m-n)}\, C|\Diamond_m\, C \in \mathrm{P}(t) \text{ y } m \geq n\}\end{aligned}$$

- o tiene dos sucesores inmediatos, s_1 y s_2, y una de las siguientes condiciones se da:
  1. $\mathrm{P}(s_1) = \mathrm{P}(t)$, $X(s_1) = 1$ y para algún H
     a) $G(s_1) \rightarrow \Diamond_p\, H \in \mathrm{P}(t)$,
     b) $\mathrm{P}(s_2) = \mathrm{P}(s) \cup \{H\}$, donde $\mathrm{P}(s)$ es el conjunto especificado en el item anterior reemplazando todas las apariciones de n por p.
     c) $X(s_2) = 1$ y $G(s_2) = \Diamond_{n-p}\, A$ si $n \geq p$ o $\Diamond_{p-n}\, A$ si $p > n$.
  2. análogo al item 1 reemplazando $G(s_1) \rightarrow \Diamond_p\, H \in \mathrm{P}(t)$ por $G(s_1) \rightarrow \Diamond_p\, H \in \mathrm{P}(t)$, se aplica la regla del espejo a $\mathrm{P}(s_2)$ y $G(s_2) = \Diamond_{n+p}\, A$.
  3. $\mathrm{P}(s_1) = \mathrm{P}(t)$, $X(s_1) = 1$ y para alguna formula $\boxplus(B \rightarrow \Diamond_p H) \in \mathrm{P}(t)$: $G(s_1) = \Diamond_q\, B$, $X(s_2) = 1$, $G(s_2) = \Diamond_{n-(p+q)}\, A$ si $n > p + q$ o $\Diamond_{(p+q)-n}\, A$ si $p + q \geq n$ y
     $$\begin{aligned}\mathrm{P}(s_2) = \quad & \{C|\Diamond_{p+q}\, C \in \mathrm{P}(t)\} \cup \{C|\boxplus C \in \mathrm{P}(t)\} \cup \{H\} \cup \\ & \{\boxplus C|\boxplus C \in \mathrm{P}(t)\} \cup \{\Diamond_{p+q} \boxminus C|\boxminus C \in \mathrm{P}(t)\} \cup \\ & \{\Diamond_{p+q}\, C|C \in \mathrm{P}(t) \text{ y } C \text{ es una cláusula ordinaria }\} \cup \\ & \{\Diamond_{((p+q)-(t+m))}\, C|\Diamond_m\, C \in \mathrm{P}(t) \text{ y } p+q > t+m\} \cup \\ & \{\Diamond_{m+((p+q)-t)}\, C|\Diamond_m\, C \in \mathrm{P}(t) \text{ y } p+q \geq t\} \cup \\ & \{\Diamond_{m-(t-(p+q))}\, C|\Diamond_m\, C \in \mathrm{P}(t) \text{ y } t > p+q\} \cup \\ & \{\Diamond_{((t+m)-(p+q))}\, C|\Diamond_m\, C \in \mathrm{P}(t) \text{ y } t+m \geq p+q\}\end{aligned}$$
  4. análogo al item 3 reemplazando $\boxplus(B \rightarrow \Diamond_p\, H) \in \mathrm{P}(t)$ por $\boxplus(B \rightarrow \Diamond_p\, H) \in \mathrm{P}(t)$ y para la construcción de $\mathrm{P}(s_2)$ y $G(s_2)$ se debe considerar que:
     - Si $q - p \geq 0$ entonces se reemplaza $p + q$ por $q - p$.
     - Si $p - q > 0$ entonces se reemplaza $p + q$ por $p - q$, se aplica la regla del espejo al conjunto $\mathrm{P}(s_2)$ y $G(s_2) = \Diamond_{n+(p-q)}\, A$.
  5. análogo al item 4 reemplazando $\boxplus(B \rightarrow \Diamond_p\, H) \in \mathrm{P}(t)$ por $\boxminus(B \rightarrow \Diamond_p\, H) \in \mathrm{P}(t)$, y para la construcción de $\mathrm{P}(s_2)$ y $G(s_2)$ se debe considerar que:
     - Si $p - q \geq 0$ entonces se reemplaza $p + q$ por $q - p$.
     - Si $q - p > 0$ entonces se reemplaza $p + q$ por $p - q$, se aplica la regla del espejo al conjunto $\mathrm{P}(s_2)$ y $G(s_2) = \Diamond_{n+(q-p)}\, A$.
  6. análogo al item 3 reemplazando $\boxplus(B \rightarrow \Diamond_p\, H) \in \mathrm{P}(t)$ por $\boxminus(B \rightarrow \Diamond H) \in \mathrm{P}(t)$, $\mathrm{P}(s_2)$ es el mismo aplicando primero la regla del espejo y $G(s_2) = \Diamond_{p+q+n}\, A$.
  7. análogo al item 3 reemplazando $\boxplus(B \rightarrow \Diamond_p\, H) \in \mathrm{P}(t)$ por $\boxplus(B \rightarrow H) \in \mathrm{P}(t)$ y para la construcción de $\mathrm{P}(s_2)$ se considera $p = 0$.
  8. análogo al item 7 reemplazando $\boxplus(B \rightarrow H) \in \mathrm{P}(t)$ por $\boxminus(B \rightarrow H) \in \mathrm{P}(t)$ aplicando la regla del espejo a $\mathrm{P}(s_2)$, $G(s_2) = \Diamond_{q+n}\, A$.

3. Modificando el uso de la negación

Los lenguajes descriptos en la sección anterior cuentan con un operador de negación $\neg$ cuya semántica fue descripta como:

$$(\sigma, j) \models \neg p \text{ si solo si } (\sigma, j) \not\models p$$

Intuitivamente esta negación es conocida como *negación por falla*, ya que significa que una fórmula $\neg A$ es **verdadera** si solo si el sistema no puede probar la veracidad de A. Esta semántica es bastante diferente de la semántica de la negación clásica, en la cual la veracidad de $\neg A$ depende solo de la prueba de la fórmula negada. La decisión de la semántica a utilizar tiene importantes implicancias para el lenguaje resultante. Si la elección es por la semántica de negación por falla, se pierden propiedades de la lógica subyacente además de algunos teoremas de la teoría. Este aspecto atenta claramente contra el uso del lenguaje en un contexto de razonamiento temporal. Por otra parte, y teniendo en cuenta la incompletitud de las especificaciones el uso de negación clásica conducirá a un sistema tri-valuado. Esto se debe a que el sistema ya no podrá responder siempre afirmativa o negativamente a una consulta; en lugar de eso la respuesta podría ser **verdadero**, **falso** o **desconocido**. En el desarrollo de los lenguajes la respuesta **desconocido** puede parecer peligrosa porque el sistema no tendría es esos casos un comportamiento previsible. Teniendo en cuenta que los lenguajes se desarrollaron para especificación, el uso de negación por falla pareció ser la solución más acertada.

Para adaptar los lenguajes a contextos de razonamiento es necesario que el sistema mantenga la mayor cantidad de propiedades de la lógica subyacente y además permita la manipulación de información negativa. Por otro lado sería prudente mantener el sistema como bi-valuado. Para lograr eso se decidió agregar a los lenguajes un operador de negación fuerte.

3.1. Extensión 1: Enfatizando la Negación por falla

Partiremos del lenguaje $EMTPL$ con el agregado de un nuevo operador ∇. Intuitivamente este operador tendría el comportamiento de la negación por falla, salvo en el caso de átomos en cuyo caso puede tomar la semántica de la negación fuerte. La sintaxis del lenguaje está dada de la siguiente manera, puede notarse la ausencia del operador $\neg$:

$$\phi = p|\phi_1 \wedge \phi_2|AT_n\phi|\nabla\phi|\oplus_n \phi|\exists n \oplus_n \phi|\forall n \oplus_n \phi|\diamondsuit_n \phi|\exists n \diamondsuit_n \phi|\forall n \diamondsuit_n \phi$$

Asumiremos nuevamente un conjunto de instantes $\ldots, i_n, i_{n+1}, i_{n+2}, \ldots$ que representan una estructura lineal e ilimitada. El sistema de información puede verse como una lista de conjuntos, σ, que contiene a los teoremas de la teoría. Cada miembro de la lista es un conjunto denotado como $\sigma(j)$. Luego $\sigma(j)$ representa el conjunto de teoremas en un determinado momento de tiempo j. Es importante tener en cuenta que (σ, j) es la aplicación del momento de tiempo j a la estructura σ dando como resultado $\sigma(j)$. La semántica del nuevo operador es:

$$(\sigma, j) \models \nabla p \text{ si solo si } \nabla p \in \sigma(j) \text{ o } (\sigma, j) \not\models p$$

Intuitivamente la semántica del operador es: "∇p es verdadera si y solo si ∇p es parte del conjunto de teoremas, o no es posible probar p". Como consecuencia la nueva definición

de $EMTPL$ solo modificaría la definición de cuerpo, remplazandose el uso de operador $\neg$ por ∇ de la siguiente manera:

- Un *cuerpo de cláusula* es una formula atómica, una conjunción de cuerpos, $\diamondsuit_n\ B$ o $\diamondsuit_n\ B$ o $\boxplus B$ o $\boxminus B$ o ∇A donde B es un cuerpo de cláusula y A un átomo.

En el algoritmo del intérprete esta modificación provocaría cambios en la regla III*a* del algoritmo original presentado en la sección 2.3 de este trabajo:

1. Si t no es un punto final y $G(t) = \nabla A$, entonces

   a) si A es un átomo q y si $X(t) = 1$ entonces $\nabla q \in \mathrm{P}(t)$ sino $\nabla q \notin \mathrm{P}(t)$ o bien

   b) t tiene exactamente un sucesor inmediato s, donde $\mathrm{P}(s) = \mathrm{P}(t)$, $X(s) = 1 - X(t)$ y $G(s) = A$

3.1.1. Ejemplo:

En extensión recién presentada, se pueden incluir reglas de la siguiente forma:

$$en\_carretera(Auto, Calle) \wedge moviendo(Auto) \ \rightarrow\ \nabla carretera\_libre(Calle)$$

En los lenguajes $MTPL$ o $EMTPL$ no se permiten reglas como la anterior ya que carece de sentido inferir algo cuyo valor de verdad depende de la falla en la prueba de otra subfórmula.

Veamos un pequeño ejemplo del aporte del operador ∇. Sea la siguiente base de conocimiento:

$$en\_calle(Auto, Calle_1) \wedge\ en\_esquina(Auto, Calle_1, Calle_2) \wedge \nabla calle\_libre(Calle2) \rightarrow \nabla permitido\_doblar(Auto, Calle_2)$$

Con el tipo de negación que cuenta el sistema, se deducirá que $Auto$ no puede doblar si efectivamente se tiene información que verifica la veracidad de que la $Calle_2$ no está libre o si no se tiene información sobre los autos que están circulando por esa calle. Este comportamiento se desprende del uso de la negación por falla. Es claro que en este ejemplo el sistema puede evitar que un determinado auto puede efectuar la maniobra de doblar en una intersección de calles aún cuando la calle este efectivamente libre.

3.2. Extensión 2: Combinando negación fuerte y por falla

Otra alternativa viable y tal vez más plausible desde el punto de vista lógico es proveer los dos tipos de negaciones simultáneamente y que, al momento de especificar el sistema, se escoja que tipo de negación es más apropiada en cada caso.

La sintaxis de los operadores en este caso sería la misma que la expresada en la extensión anterior. Veamos la semántica de los operadores de negación involucrados:

$$(\sigma, j) \models \nabla p \text{ si solo si } \nabla p \in \sigma(j) \text{ o } (\sigma, j) \models \neg p$$
$$(\sigma, j) \models \neg p \text{ si solo si } \ p \notin \sigma(j)$$

Básicamente la semántica de ∇ es la misma que en el caso anterior, solo que ahora se cuenta explícitamente con un operador de negación por falla.

En el algoritmo del intérprete esta modificación provocaría cambios nuevamente el la regla III*a* del algoritmo dado en la sección 2.3. Se debería reemplazar por las siguientes reglas:

1. Si t no es un punto final y $G(t) = \nabla A$, entonces

   a) si A es un átomo q y si $X(t) = 1$ entonces $\nabla q \in \mathrm{P}(t)$ sino $\nabla q \notin \mathrm{P}(t)$ o bien

   b) t tiene exactamente un sucesor inmediato s, donde $\mathrm{P}(s) = \mathrm{P}(t)$, $X(s) = 1 - X(t)$ y $G(s) = A$

2. Si t no es un punto final y $G(t) = \nabla A$, entonces

   a) si A es un átomo q y si $X(t) = 1$ entonces $\nabla q \in \mathrm{P}(t)$ sino $\nabla q \notin \mathrm{P}(t)$ o bien

   b) t tiene exactamente un sucesor inmediato s, donde $\mathrm{P}(s) = \mathrm{P}(t)$, $X(s) = 1$ y $G(s) = \neg A$

3. Si t no es un punto final y $G(t) = \neg A$, entonces t tiene exactamente un sucesor inmediato s, donde $\mathrm{P}(s) = \mathrm{P}(t)$, $X(s) = 1 - X(t)$ y $G(s) = A$

3.2.1. Ejemplo:

En la extensión recién presentada, se pueden combinar ambas formas de negación. Se mantiene la capacidad de inferir información negativa como en el caso anterior. La principal diferencia que se tiene con respecto a la primer extensión presentada radica en que si en el caso de la regla dada anteriormente:

$$en\_calle(Auto, Calle_1) \wedge\ en\_esquina(Auto, Calle_1, Calle_2) \wedge \nabla calle\_libre(Calle2) \rightarrow \nabla permitido\_doblar(Auto, Calle_2)$$

no nos interesa probar que efectivamente la calle esta libre, sino que nos alcanza con una meta menos fuerte como garantizar que con la información de la base no se puede probar que la calle está ocupada, se puede hacer efectivamente utilizando el operador de negación por falla solo. Es decir, se obtiene la siguiente regla:

$$en\_calle(Auto, Calle_1) \wedge\ en\_esquina(Auto, Calle_1, Calle_2) \wedge \neg calle\_libre(Calle2) \rightarrow \nabla permitido\_doblar(Auto, Calle_2)$$

Es claro que eliminando esta capacidad, las dos extensiones presentadas hasta el momento se comportan de manera análoga.

3.3. Extensión 3: Solo negación fuerte

La última modificación sugerida es hacer uso de la negación tradicional, exigiendo que la especificación sea completa o permitiendo como respuesta **desconocido** a fin de que en el chequeo del modelo se conozca la falta de especificidad del mismo. Es importante tener en cuenta que en este caso el lenguaje no podría aplicarse a sistemas de tiempo real debido a la posibilidad de fallas.

En este caso la sintaxis sería la misma que la presentada en la primer extensión, solo que la semántica del operador de negación estaría dada de la siguiente manera:

$$(\sigma, j) \models \nabla p \text{ si solo si } \nabla p \in \sigma(j)$$

Se asumen nuevamente las mismas restricciones sobre la estructura temporal indicadas en las extensiones previas. Intuitivamente la negación de una subfórmula es verdadera si puede probarse fehacientemente la misma.

Para que no se pierdan propiedades de la lógica tradicional y a fin de brindar mayor libertad se podría permitir el uso del operador de negación no solo sobre átomos. De esta manera nuestra definición por ejemplo para $MTPL$ contaría con las mismas modificaciones indicadas en nuestra primer sugerencia.

Presentamos solamente las reglas relacionadas con el operador de negación en $MTPL$. En caso de considerar el lenguaje $EMTPL$ la regla en cuestión es completamente análoga. ALGORITMO (adaptado de [Gab87]) $(T, \leq, 0, V)$ es un árbol de computación etiquetado para P?$G = x$ si solo si se satisfacen las siguientes condiciones:

1. Si t no es un punto final y $G(t) = \nabla A$, entonces:

   Si A es un átomo q
   entonces si $X(t) = 1$ entonces $\nabla q \in$ P(t) sino $\nabla q \notin$ P(t)
   sino Si A es una conjunción, $B \wedge C$
   entonces t tiene exactamente dos sucesores inmediato s_1 y s_2, donde
   P(s_i) = P(t), $X(s_i) = X(t)$, $G(s_1) = \nabla B$ y $G(s_2) = \nabla C$
   sino Si A tiene la forma ∇B
   entonces t tiene exactamente un sucesor inmediato s,
   donde P(s) = P(t), $X(s) = X(t)$ y $G(s) = B$
   sino [A tiene la forma $\Diamond\!\!\!\!-B$ (o $\Diamond B$)]
   t tiene exactamente un sucesor inmediato s, donde
   P(s) = P(t), $X(s) = X(t)$ y $G(s) = \boxplus\nabla B$ ($G(s) = \boxminus\nabla B$)

3.3.1. Ejemplo:

Retomando el ejemplo presentado en las secciones anteriores, tenemos que el significado de la regla:

$$en\_calle(Auto, Calle_1) \wedge\ en\_esquina(Auto, Calle_1, Calle_2) \wedge \nabla calle\_libre(Calle2) \rightarrow \nabla permitido\_doblar(Auto, Calle_2)$$

es mucho más estricto. Solo se alcanzará el consecuente con la prueba fehaciente del antecedente, es decir, se debe probar la negación. En caso de no contar con información el sistema no verificará el antecedente.

4. Conclusiones y trabajo futuro

Se han mostrado sugerencias para agregar la capacidad de manejo de información negativa a los lenguajes $MTPL$ y $EMTPL$ a fin de que los mismos no pierdan tantas propiedades de la lógica y puedan ser considerados lenguajes aptos para el razonamiento temporal. Las tres extensiones parten de la necesidad de agregar el operador de negación clásica al sistema. Estas versiones son preliminares por lo cual aún queda pendiente el estudio de complejidad de las alternativas. Otro aspecto relacionado que merece destacarse, es que el sistema con la tercera extensión se comportará como tri-valuado solamente si la regla así lo indica, cosa que no fue indicada en las reglas de los algoritmos presentadas, ya que de otra forma si el intérprete se implementa por ejemplo sobre una base Prolog, el sistema continuará siendo bi-valuado.

Los algoritmos completos se pueden obtener reemplazando las reglas presentadas en el presente trabajo por aquellas que correspondan de los intérpretes de $EMTPL$ [CA99] y/o $MTPL$ [CA00].

Sería interesante poder contar con mecanismos de resolución como el desarrollado por Michael Fisher [Fis91] como así también por Clare Dixon [Dix95, Dix96] para lógicas no métricas. En un principio da la impresión de que las fórmulas pueden reescribirse utilizando los operadores $\oplus$ (en el próximo instante de tiempo) y $\ominus$ (en el instante previo) que Fisher agregó a su sistema aunque en algunos casos no es posible reescribirlas. Cabe destacar a si mismo que existen otros desarrollos sobre lógicas como LTL [Pnu81, MP92], la principal diferencia radica en la lógica utilizada y su naturaleza. Otro aspecto que sería interesante analizar es la posibilidad de utilizar técnicas de Model Checking a fin de mejorar aspectos de eficiencia en los algoritmos dados [CGP99].

Referencias

[CA99] M. L. Cobo and J. C. Augusto. EMTPL: A Pogramming Language for Temporal Deductive Data Bases. In *Proceedings de la* XIX International Conference of the Chilean Computer Science Society, pages 170–178, Talca, Chile, 1999.

[CA00] M. L. Cobo and J. C. Augusto. Towards a Programming Language Based on Prior's Metric Temporal Operators. In *Proceedings del VI Congreso Argentino de Cs. de la Computación, CACiC2000*, pages 453 –464, 2000.

[CGP99] E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. 1999.

[Cob01] M. L. Cobo. Fundamentos de la programación en lógica temporal métrica. 2001.

[Dix95] C. Dixon. *Strategies for Temporal Resolution*. PhD thesis, University og Manchester, Deparment of Computer Science, Manchester, United Kingdom, 1995.

[Dix96] C. Dixon. Temporal resolution: A breadth-first approach. In *TIME-96 the Third International Workshop on Temporal Representation and Reasoning*, pages 120–127, 1996.

[Fis91] M. Fisher. A resolution method for temporal logic. In *Twelfth Intenational Joint Conference on Artifitial Intelligence (IJCAI)*, pages 99–104, 1991.

[Gab87] D. M. Gabbay. Modal and temporal logic programming. In Antony Galton, editor, *Temporal Logic and their Applications*, pages 197–236. Academic Press, 1987.

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. 1992.

[Pnu81] A. Pnueli. The temporal logic of reactive and concurrent programs. *Theoretical Computer Science*, 13:45–60, 1981.

[Pri67a] A. Prior. *Past, Present and Future*. Clarendon Press, 1967.

[Pri67b] A. Prior. Stratified metric tense logic. *Theoria 33*, pages 28–38, 1967.

[Vil94] L. Vila. A survey on temporal reasoning in artificial intelligence. *AI Communications*, 7(1):4–28, 1994.